In the Claims:

Please amend claims 13, 18, 21-43, 46, 47, 52, 54, and 56, and please cancel claims 44-45, as indicated below.

1. (Original) A method of providing non-blocking multi-target transactions in a computer system, the method comprising:

defining plural transactionable locations, wherein individual ones of the transactionable locations encode respective values and are owned by no more than one transaction at any given point in a multithreaded computation;

for a particular multi-target transaction of the multithreaded computation, attempting to acquire ownership of each of the transactionable locations targeted thereby, wherein the ownership acquiring wrests ownership from another transaction, if any, that owns the targeted transactionable location; and

once ownership of each of the targeted transactionable locations has been acquired, attempting to commit the particular multi-target transaction using a single-target synchronization primitive to ensure that, at the commit, the particular multi-target transaction continues to own each of the targeted transactionable locations, wherein individual ones of the multi-target transactions do not contribute to progress of another.

2. (Previously presented) The method of claim 1,

wherein the ownership wresting employs a single-target synchronization primitive to change status of the wrested-from transaction to be incompatible with a commit thereof.

3. (Previously presented) The method of claim 2,

wherein, as a result of the status change, the wrested-from transaction fails and retries.

4. (Previously presented) The method of claim 2,

wherein the wrested-from transaction is itself a multi-target transaction.

5. (Original) The method of claim 1, further comprising:

on failure of the commit attempt, reacquiring ownership of each targeted transactionable location and retrying.

6. (Original) The method of claim 1,

wherein no transaction may prevent another from wresting therefrom ownership of transactionable locations targeted by the active transaction.

7. (Original) The method of claim 1,

wherein the ownership acquiring employs a single-target synchronization primitive to update the ownership of the targeted transactionable location.

8. (Original) The method of claim 1,

wherein each encoding of a transactionable location is atomically updateable using a single-target synchronization primitive.

9. (Original) The method of claim 1,

wherein the individual transactionable location encodings further include an identification of the owning transaction's corresponding value for the transactionable location.

10. (Original) The method of claim 1, further comprising:

accessing values corresponding to individual ones of the transactionable locations using a wait-free load operation.

11. (Original) The method of claim 1,
wherein the transactionable locations directly encode the respective values.

12. (Original) The method of claim 1,
wherein the transactionable locations are indirectly referenced.

13. (Currently amended) The method of claim 1,
wherein the transactionable locations are encoded in storage managed using a non-blocking memory management technique.

14. (Original) The method of claim 1,
wherein the transactionable locations, if unowned, directly encode the respective values and otherwise encode a reference to the owning transaction.

15. (Original) The method of claim 1,
wherein the single-target synchronization primitive employs a Compare-And-Swap (CAS) operation.

16. (Previously presented) The method of claim 1,
wherein the single-target synchronization primitive employs a Load-Linked (LL) and Store-Conditional (SC) operation pair.

17. (Original) The method of claim 1,
wherein the single-target of the single-target synchronization primitive includes at least a value and a transaction identifier encoded integrally therewith.

18. (Currently amended) The method of claim 1,
wherein the multi-target transaction ~~has semantics of~~ comprises a multi-target compare and swap (NCAS) operation.

19. (Previously presented) The method of claim 1,

embodied in operation of an application programming interface (API) that includes a load operation and a multi-target compare and swap (NCAS) operation.

20. (Original) The method of claim 19,

wherein the load operation is wait-free.

21. (Currently amended) The method of claim 1,

embodied in operation of an application programming interface (API) that provides transactional memory functionality.

22. (Currently amended) ~~An implementation of~~ A computer-readable storage medium storing program instructions computer-executable to implement:

a plurality of non-blocking, multi-target transactions;

wherein the program instructions comprise:

~~that employs~~ instances of one or more single-target synchronization primitives executable to acquire, for a particular multi-target transaction, ownership of targeted transactionable locations; and

a particular single-target synchronization primitive executable to ensure that, at commit, the particular multi-target transaction continues to own each of the targeted transactionable locations[[,]]; and

wherein individual ones of the multi-target transactions do not contribute to progress of others.

23. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein the program instructions are further executable to implement ~~embodied as software encoded in one or more computer readable media and that, on execution as part of~~ a concurrent computation, and wherein execution of the concurrent computation invokes the multi-target transactions.

24. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein ~~the ownership acquiring~~ to acquire ownership, ~~when performed by~~ a first one of the multi-target transactions[[,]] wrests ownership from respective other ones of the multi-target transactions, if any, that own respective ones of the targeted transactionable locations.

25. (Currently amended) The ~~implementation~~ storage medium of claim 24, wherein ~~the wresting employs~~ to wrest ownership, the program instructions are further executable to implement an instance of a single-target synchronization primitive [[to]] ~~change~~ changing status of a wrested-from transaction to be incompatible with a commit thereof.

26. (Currently amended) The ~~implementation~~ storage medium of claim 25, wherein, as a result of the status change, the program instructions are further executable to implement the wrested-from transaction eventually ~~fails~~ failing and ~~retries~~ retrying.

27. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein no transaction may prevent another from wresting therefrom ownership of transactionable locations targeted by the active transaction.

28. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein the transactionable locations directly encode [[the]] respective values.

29. (Currently amended) The ~~implementation~~ storage medium of claim 22,

wherein the transactionable locations are indirectly referenced.

30. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein the transactionable locations are encoded in storage managed using a non-blocking memory management technique.

31. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein the transactionable locations, if unowned, directly encode [[the]] respective values and otherwise encode a reference to [[the]] an owning transaction.

32. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein at least some instances of the one or more single-target synchronization primitives employ a Compare-And-Swap (CAS) operation.

33. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein at least some instances of the one or more single-target synchronization primitives employ a Load-Linked (LL) and Store-Conditional (SC) operation pair.

34. (Currently amended) The ~~implementation~~ storage medium of claim 22, wherein at least some of the multi-target transactions ~~have semantics of~~ comprise a multi-target compare and swap (NCAS) operation.

35. (Currently amended) The ~~implementation~~ storage medium of claim 22, ~~embodied as software that includes a functional encoding of~~ wherein the program instructions comprise operations concurrently executable by one or more processors to operate on state of the transactionable locations.

36. (Currently amended) The ~~implementation~~ storage medium of claim 22,

wherein at least some of the multi-target transactions are defined by an application programming interface (API) that includes a load operation and a multi-target compare and swap (NCAS) operation.

37. (Currently amended) The ~~implementation~~ storage medium of claim 22,
wherein at least some of the multi-target transactions are defined by an application programming interface (API) that provides transactional memory functionality.

38. (Currently amended) The ~~implementation~~ storage medium of claim 22,
wherein the multi-target transactions are obstruction-free, though not wait-free or lock-free.

39. (Currently amended) The ~~implementation~~ storage medium of claim 22,
wherein the ~~implementation~~ program instructions do[[es]] not ~~itself~~ guarantee that at least one interfering concurrently executed multi-target transactions makes progress.

40. (Currently amended) The ~~implementation~~ storage medium of claim 22,
wherein the program instructions are further executable to implement a contention management facility ~~is employed~~ configured to facilitate progress in a concurrent computation.

41. (Currently amended) The ~~implementation~~ storage medium of claim 40,
wherein ~~operation~~ execution of the contention management facility ensures progress of the concurrent computation.

42. (Currently amended) The ~~implementation~~ storage medium of claim 40,
wherein the contention management facility is modular such that alternative contention management strategies may be employed without affecting correctness ~~of the implementation~~.

43. (Currently amended) The ~~implementation~~ storage medium of claim 40, wherein the contention management facility allows changes in contention management strategy during a course of the concurrent computation.

44. (Cancelled)

45. (Cancelled)

46. (Currently amended) A computer readable storage medium ~~encoding~~ storing program instructions computer-executable to implement ~~at least a portion of a nonblocking, multi-target transaction implementation, the encoding comprising~~:

~~a definition~~ instantiation of [[a]] one or more transactionable locations ~~instantiable~~ in shared memory configured to individually encapsulate values that may be targeted by concurrent executions of [[the]] non-blocking multi-target transactions; and

~~a functional encoding of the~~ one or more instances of a non-blocking multi-target transaction that upon execution of a particular instance thereof, attempts to acquire ownership of each of [[the]] a plurality of transactionable locations targeted thereby and, once ownership of each of the targeted transactionable locations has been acquired, attempts to commit the particular instance using a single-target synchronization primitive to ensure that, at the commit, the particular instance continues to own each of the targeted transactionable locations, wherein execution of no one of the multi-target transaction instances contributes to progress of another.

47. (Currently amended) The storage medium of claim 46,

wherein the ownership acquiring wrests ownership from another transaction, if any, that owns one of the targeted transactionable locations.

48. (Previously presented) The storage medium of claim 47,
wherein the another transaction is another concurrently executing instance of the multi-target transaction.

49. (Previously presented) The storage medium of claim 46,
wherein at least some instances of the single-target synchronization primitive employ a Compare-And-Swap (CAS) operation.

50. (Previously presented) The storage medium of claim 46,
wherein at least some instances of the single-target synchronization primitive employ a Load-Linked (LL) and Store-Conditional (SC) operation pair.

51. (Previously presented) The storage medium of claim 46,
wherein the single-target of the single-target synchronization primitive includes a value and an owning transaction identifier encoded integrally therewith.

52. (Currently amended) The storage medium of claim 46,
wherein the program instructions are embodied as an application programming interface software component combinable with application program code to facilitate execution of the application program code as a multithreaded computation.

53. (Previously presented) The storage medium of claim 46,
wherein the multi-target transaction implements a multi-target compare and swap operation (NCAS).

54. (Currently amended) The storage medium of claim 46,

wherein the multi-target transaction implements transactional memory functionality.

55. (Previously presented) The storage medium of claim 46,

wherein the computer readable storage medium includes at least one medium selected from the set of a disk, a tape and another magnetic, optical, or electronic storage medium.

56. (Currently amended) An apparatus, comprising:

one or more processors;

one or more data stores addressable by each of the one or more processors; and

means for coordinating concurrent non-blocking execution, by the one or more processors, of multi-target transactions that attempt to acquire ownership of each of a plurality of transactionable locations targeted thereby and, once ownership of each of the targeted transactionable locations has been acquired, attempt to commit [[the]] a particular instance thereof using a single-target synchronization primitive to ensure that, at the commit, the particular instance continues to own each of the targeted transactionable locations, wherein none of the multi-target transaction contributes to progress of another.

57. (Original) The apparatus of claim 56, further comprising:

means for wresting ownership from another transaction, if any, that owns one of the targeted transactionable locations.

58. (Previously presented) The apparatus of claim 56,

wherein the wresting means includes means for ensuring that status of the wrested-from transaction is incompatible with a successful commit thereof.

59. (Original) The apparatus of claim 56, further comprising:

means for managing contention between interfering executions of the multi-target transactions.